LinearReg

Gerardo De la O

2023-01-10

Question 8.1

The first company I worked for specialized in commercial energy management via IoT devices (smart buildings). Every site we managed contained a plethora of equipment used for every day operations. One of the best use cases of linear regression for owners and utility managers is to predict the energy consumption of a given site. Being able to approximate energy consumption enables management to allocate the correct amount of funds for utilities and while at the same time implementing practices to save energy.

The following predictors would be useful to compute a good energy consumption model:

1. Avg. outside temperature
2. Type of building and square footage
3. Occupied hours
4. Type of heating (electricity/gas)
5. Equipement counts (AC Units, Lights, Refrigerators, etc.)

The nice part about modeling electricity usage is that you can evaluate the accuracy of your prediction every single month via utility bills! Therefore, the model would be able to be refined over time.

Question 8.2

This section explores the use of Linear Regression to predict crime rate in a city, given a set of parameters. The following sections illustrate the model building process.

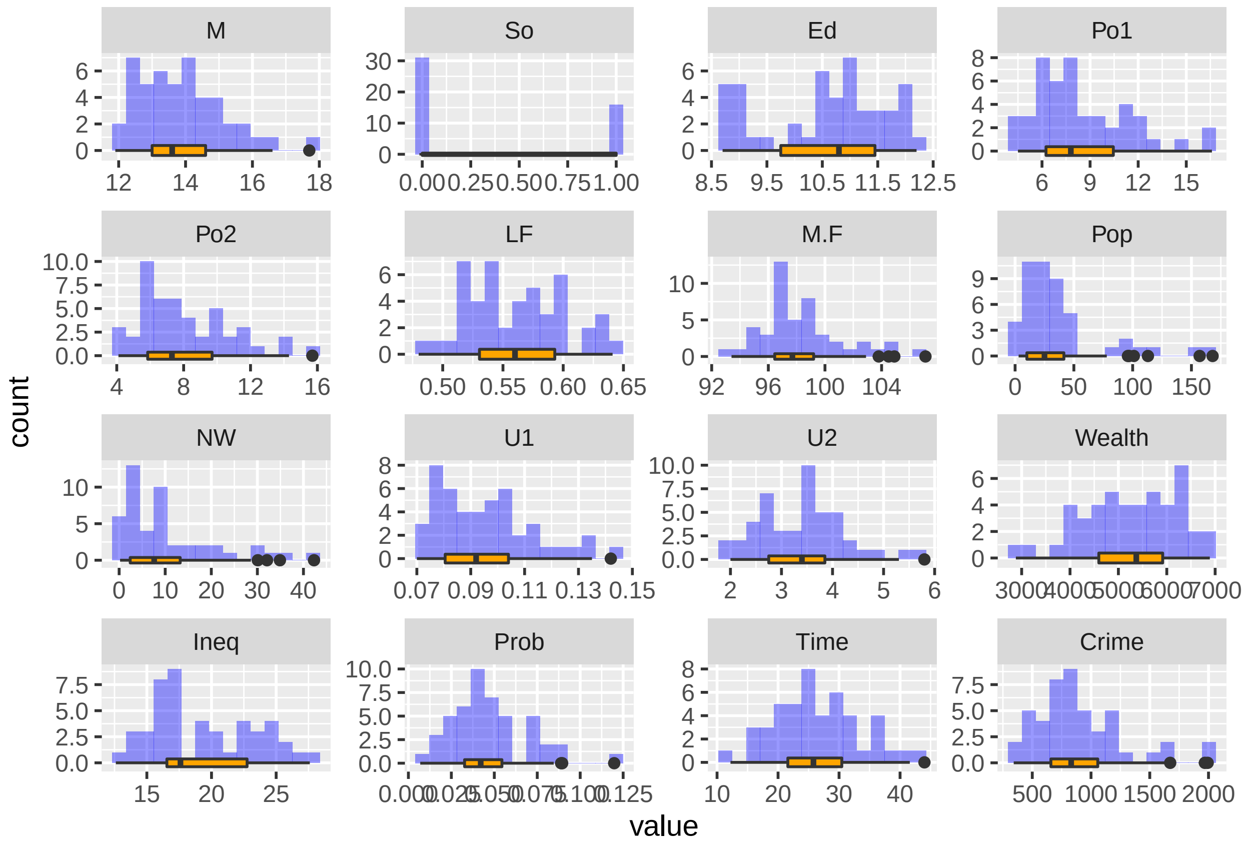
Preparation

The purpose of this step is to understand the data and perform some basic cleanup. The graph below shows that we have 1 binary and 14 continous predictors. Also, the continuous features all seem to be approximately normal. This is good news, as we don't have to create dummy variables or perform any column manipulations. The scales do seem different for some of the features, and there appear to be a few outliers on a few of the features. So we perform the following steps:

1. Remove rows where a z-score is greater than 3 (outside the 99.7th percentile). In his case, we favor a model that behaves well within its bounds over something super general.
2. Standardize the data to $\mu = 0$ and sd $= 1$

```
library(tidyverse)
library(reshape2)
library(lubridate)
library(caret)

#import data
filename = 'http://www.statsci.org/data/general/uscrime.txt'
data <- read.csv(filename, sep = '\t' ) #tab delimited
```

```
# Melt to plot easily
melted_data <- melt(data)

ggplot(data=melted_data, aes(x = value)) +
    geom_histogram(bins=15, fill='blue', alpha=0.4) +
    geom_boxplot(fill='orange') +
    facet_wrap(~variable, scales = "free")
```



```
# Outlier removal
z_scores <- as.data.frame(sapply(data, function(data) (abs(data-mean(data))/sd(data))))
delete = rowSums(z_scores > 3)

df <- data[!delete, ]
dim(df)
```

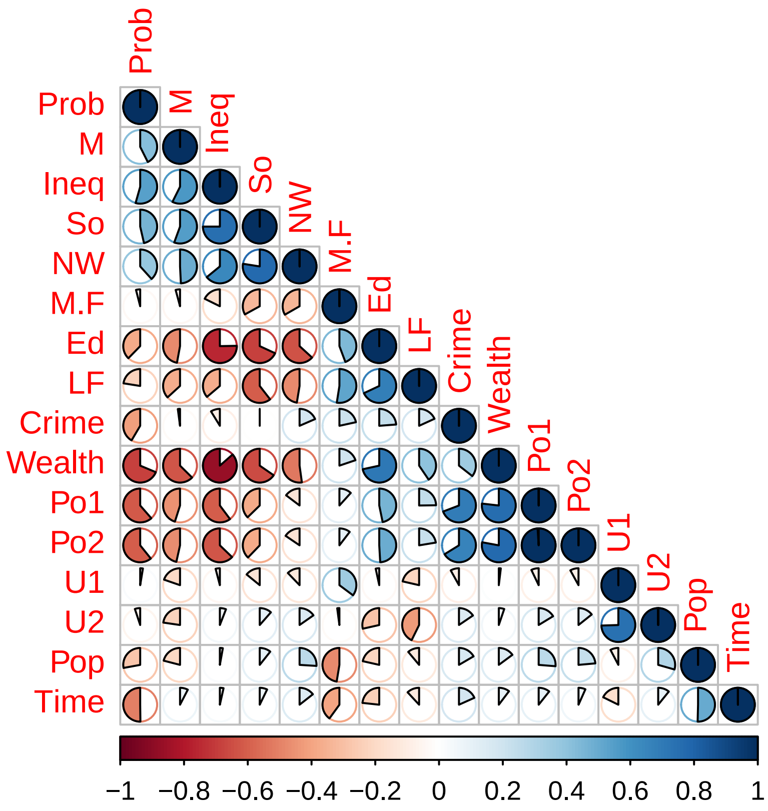
```
## [1] 42 16
```

```
# Scale the data
stdize = preProcess(df[,-16], method = c("center", "scale"))
df_sc = round(predict(stdize, df),4)

# Scale  and store the given input vector
testVal  = df_sc[FALSE,-16]
testVal[nrow(testVal) + 1,] = c(14,0,10,12,15.5,.640,94,150,1.1,.120,3.6,3200,20.1,.04,39)
testVal_sc = predict(stdize, testVal)
```

Correlation Analysis

Since we are building a linear regression model, we need to make sure a few conditions are met before we can draw inferences about the estimates. We have visually confirmed that our distributions are approximately normal, but another thing we can check is the presence of multicollinearity. To test this, we compute the correlations of all variables.

```r
library(corrplot)

# Check out Pearson correlations
corr <- round(cor(df_sc, method='pearson'),3)
corrplot(corr, method='pie', type='lower', order='hclust', insig = "blank")
```



From the plot, we see quite a few high correlations. The most obvious one is Po1 with Po2, so we remove Po2 to keep the most recent factor. In practice, we would probably do a more careful feature selection since the amount of correlation is very high. But our single variable removal illustrates the method.

Analysis

We are finally ready to fit the model. The model uses all variables except Po2, which turns out results in a lower test set error than using all of the variables!

```r
set.seed(1)


# input = df_sc
input = subset(df_sc, select=-c(Po2))
```

```r
# generate train/test sets
smp_size <- floor(.80 * nrow(input))
sampler <- sample(seq_len(nrow(input)), size = smp_size)
train <- input[sampler, ]
test <- input[-sampler, ]

# train  model
model <- lm(Crime ~ .,data=train)
fit = model$fitted.values

# Fit model to test set
testRes = predict(model, test)

# Compute mean absolute error on test set
MAE = sum(abs(test$Crime - testRes))/nrow(test)
MAE
```

```
## [1] 239.4432
```

8.2 Answer

Model & Quality of Fit The output below shows the model coefficients along with errors, statistic, P values, and a few other evaluation criteria. It is worth noting that not all of the variables are statistically significant, suggesting the possibility of further refining the model.

```r
library (broom)

output <- tidy(model)
print(summary(model))
```

```
##
## Call:
## lm(formula = Crime ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -229.70  -71.06  -17.30   87.85  333.04
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   914.11      31.62  28.909  < 2e-16 ***
## M             102.27      53.57   1.909  0.07235 .
## So            -39.69      76.76  -0.517  0.61141
## Ed            285.03      76.95   3.704  0.00162 **
## Po1           256.16      69.39   3.692  0.00167 **
## LF            -92.56      87.07  -1.063  0.30177
## M.F            93.10      74.31   1.253  0.22626
## Pop            32.42      51.58   0.628  0.53758
## NW             72.16      58.54   1.233  0.23354
## U1           -161.31      90.52  -1.782  0.09160 .
## U2            176.98      79.27   2.233  0.03851 *
## Wealth        -62.47      95.28  -0.656  0.52033
## Ineq          193.08      92.65   2.084  0.05169 .
## Prob          -97.08      68.37  -1.420  0.17272
## Time           20.58      59.99   0.343  0.73550
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 173.8 on 18 degrees of freedom
## Multiple R-squared:  0.8927, Adjusted R-squared:  0.8092
## F-statistic: 10.69 on 14 and 18 DF,  p-value: 5.148e-06
```

Overall, the fit seems very good, with an r^2 value of .89! Given the amount of data, number of features, and the test MSE value of 239, we suspect a degree of over-fitting. We can check for and correct over-fitting using various methods outside the scope of this assignment.

Prediction Finally, we run our model with the given data set! The model returns a predicted crime rate of 1026.9, which is above average, but still within normal range.

```
predict(model, newdata=testVal_sc)
```

```
##        1
## 1026.898
```